REMARKS

Claims 23-34 have been canceled. New claims 35-45 have been added. Claims 35-45 remain pending.

Applicants traverse the Examiner's rejection of the claims under 35 USC §112, paragraph nos. 1 and 2. The claims have been canceled and replaced with new claims 35-45. New claims 35-45 correspond more closely to the original claims. However, translation errors (such as "programme") have been corrected. The fact that new specific instructions are generated in step c) is highlighted and the operation of the program language interpreter when the compacted program is run is also highlighted in the independent claims 35, 38 and 41. Moreover, claim 38 has been redrafted as an independent claim, not dependent on claim 35.

Applicants further traverse the rejection of claims 23-34 as being anticipated by SISKA (U.S. Patent No. 6,263,429) under 35 USC 102(e).

The present invention relates to on-board systems in which a minimization of the necessary memory resources is scarched. In this context, the program language interpreter (recited in independent claims 35, 38 and 41) is an important means to that end, as explained hereafter.

The method of compacting according to the present application is performed on an intermediate program. Initially, the intermediate program consists on series of code values which are the "*standard instructions*" as recited in the claims (see for example page 8, lines 6-13 of the application as originally filed). When the intermediate program is run, the microprocessor reads the code values through the interpreter. The interpreter refers to a table of standard instructions TAB-STD (see figure 2c) in order to interpret the read code values. Therefore, the

microprocessor can read the standard instructions and the interpreter finds in the table TAB-STD the information for running these standard instructions (see page 13, line 25).

When the compaction process according to the invention is performed on the intermediate program, same sequences of successive code values ("*standard instructions*" as recited in the claims) are searched and replaced by a single code value ("specific instructions" as recited in the claims). It should be noted that this single code value is a new one. In other terms, the specific instructions are new ones, compared to the standard instructions. The specific instructions are created during the compacting process (see step c) of claim 35).

A clear example is given the specification (page 11, line 3 – page 12, line 6). In that example, the intermediate program consists on the following code values (standard instructions):

1 - 7 - 3 - 5 - 7 - 3 - 7 - 3 - 5 - 7

A same sequence of four code values is recognized twice: [7 - 3 - 5 - 7].

A new code value (for example 106) is assigned to [7 - 3 - 5 - 7].

Therefore the series: 1 - (7 - 3 - 5 - 7) - 3 - (7 - 3 - 5 - 7] becomes:

1 - 106 -3- 106

(see page 12, line 6).

Referring to page 11, lines 18-29 of the specification of the application, standard code values can be for example from 000 to 105, while specific code values can be 106, 107, 108, etc.

Thus, it should be understood that:

- new specific instructions are created during the compaction process, and
- the language program interpreter can interpret the new specific instructions exactly like mere standard instructions.

Such an interpreter is not disclosed in SISKA. Indeed, it cannot be disclosed in SISKA since SISKA does not create any new instruction. The microcalls of SISKA are not new instructions, since they can be read by the microprocessor without the use of an interpreter.

In the invention, the interpreter ensures the software adaptation of the standard and specific instructions of the compacted intermediate program into instructions which can be run directly by the microprocessor. This is clearly indicated in page 6, lines 17-19 of the application as filed: "*the intermediate object code program consists on a series of standard instructions which can be run by the microprocessor through the interpreter.*" Therefore, the intermediate program (or the compacted intermediate program) can be run by the microprocessor only through the interpreter. As a matter of fact, if a new "*specific*" instruction does not exist prior to its interpretation by the interpreter, it cannot be read by the microprocessor. On the opposite, in SISKA, a "microcall" is a simple instruction that the microprocessor already knows.

Moreover, SISKA adds that: "*A processor can identify when to excite a microroutine and return to the program*" (col. 10, lines 51-53). A microcall of SISKA (col. 9, line 67) calls a routine which is external to the program itself. On the opposite, in an embodiment of the present invention, the *standard* instructions and the *specific* instructions are comprised in a same stack as recited in claim 40 of the new set of claims herewith. This feature cannot be retrieved in SISKA.

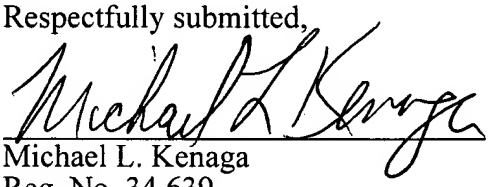
Still further, the Examiner's attention is directed to the use of a table of standard instructions TAB-STD (as recited in claim 41), so that the interpreter of the invention cannot be disclosed in SISKA. It can be pointed out that this feature has never been taken into account during the examination, while it is recited in claim 7 as originally filed. Moreover, the fact that the standard instructions and the specific instructions can be within a same stack is a feature that

has never been taken into account during the examination while it was recited in claim 6 as originally filed.

In addition, applicants are finalizing a declaration under 37 CFR §1.131 establishing an invention date prior to the effective date of SISKA. The declaration will be submitted shortly.

Favorable consideration and prompt allowance of the application are respectfully requested.

Respectfully submitted,



Michael L. Kenaga
Reg. No. 34,639

DLA PIPER RUDNICK GRAY CARY US LLP
P.O. Box 64807
Chicago, Illinois 60664-0807
Phone: 312/368-4000
Customer No.: 28465